

Lecture 8, 9

The Relational Data Model and Relational Database Constraints

Relational Database Design by ER- and EERR-to-Relational Mapping

Lecture Outline

- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas
- ER-to-Relational Mapping Algorithm
- Mapping EER Model Constructs to Relations

Relational Model Concepts

- The relational Model of Data is based on the concept of a *Relation*
 - The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations
- We review the essentials of the *formal relational model* in this chapter
- A Relation is a mathematical concept based on the ideas of sets

Informal Definitions

- Informally, a **relation** looks like a **table** of values.
- A relation typically contains a **set of rows**.
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
 - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
 - In the formal model, the column header is called an **attribute name** (or just **attribute**)

Example of a Relation

Relation Name

STUDENT

Attributes

Tuples

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1

The attributes and tuples of a relation STUDENT.

Informal Definitions

- Key of a Relation:
 - Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
 - Called the *key*
 - In the STUDENT table, SSN is the key
- Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table
 - Called *artificial key* or *surrogate key*

Formal Definitions - Schema

- The **Schema** (or description) of a Relation:
 - Denoted by $R(A_1, A_2, \dots, A_n)$
 - R is the **name** of the relation
 - The **attributes** of the relation are $A_1, A_2, \dots, A_n$
- Example:
CUSTOMER (Cust-id, Cust-name, Address, Phone#)
 - CUSTOMER is the relation name
 - Defined over the four attributes: Cust-id, Cust-name, Address, Phone#
- Each attribute has a **domain** or a set of valid values.
 - For example, the domain of Cust-id is 6 digit numbers.

Formal Definitions - Summary

- Formally,
 - Given $R(A_1, A_2, \dots, A_n)$
 - $r(R) \subset \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$
- $R(A_1, A_2, \dots, A_n)$ is the **schema** of the relation
- R is the **name** of the relation
- $A_1, A_2, \dots, A_n$ are the **attributes** of the relation
- $r(R)$: a specific **state** (or "value" or "population") of relation R – this is a *set of tuples* (rows)
 - $r(R) = \{t_1, t_2, \dots, t_n\}$ where each t_i is an n -tuple
 - $t_i = \langle v_1, v_2, \dots, v_n \rangle$ where each v_j *element-of* $\text{dom}(A_j)$

Definition Summary

<u>Informal Terms</u>		<u>Formal Terms</u>
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Example – A relation STUDENT

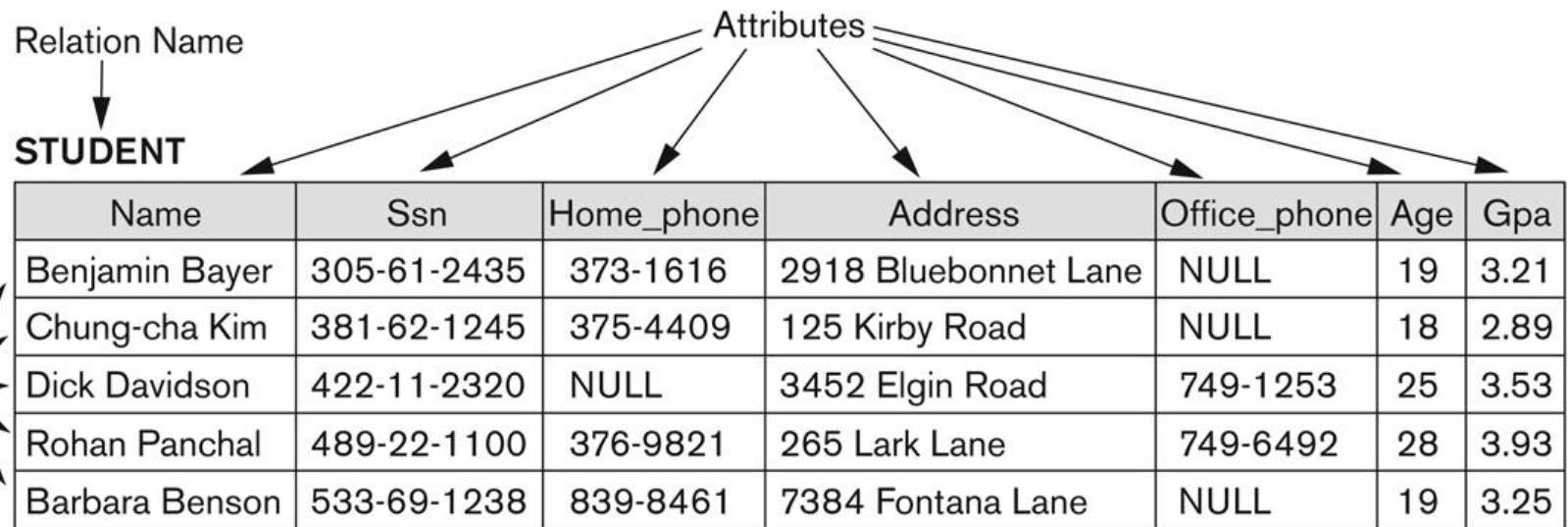


Figure 5.1

The attributes and tuples of a relation STUDENT.

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- There are three *main types* of (explicit schema-based) constraints that can be expressed in the relational model:
 - **Key** constraints
 - **Entity integrity** constraints
 - **Referential integrity** constraints
- Another schema-based constraint is the **domain** constraint
 - Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

Key Constraints

- **Superkey of R:**
 - Is a set of attributes SK of R with the following condition:
 - No two tuples in any valid relation state $r(R)$ will have the same value for SK
 - That is, for any distinct tuples t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$
 - This condition must hold in *any valid state* $r(R)$
- **Key of R:**
 - A "minimal" superkey
 - That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)
- A Key is a Superkey but not vice versa

Key Constraints (continued)

- Example: Consider the CAR relation schema:
 - CAR(State, Reg#, SerialNo, Make, Model, Year)
 - CAR has two keys:
 - Key1 = {State, Reg#}
 - Key2 = {SerialNo}
 - Both are also superkeys of CAR
 - {SerialNo, Make} is a superkey but *not* a key.
- In general:
 - Any *key* is a *superkey* (but not vice versa)
 - Any set of attributes that *includes a key* is a *superkey*
 - A *minimal* superkey is also a key

Key Constraints (continued)

- If a relation has several **candidate keys**, one is chosen arbitrarily to be the **primary key**.
 - The primary key attributes are underlined.
- Example: Consider the CAR relation schema:
 - CAR(State, Reg#, SerialNo, Make, Model, Year)
 - We chose SerialNo as the primary key
- The primary key value is used to *uniquely identify* each tuple in a relation
 - Provides the tuple identity
- Also used to *reference* the tuple from another tuple
 - General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
 - Not always applicable – choice is sometimes subjective

CAR table with two candidate keys – LicenseNumber chosen as Primary Key

CAR

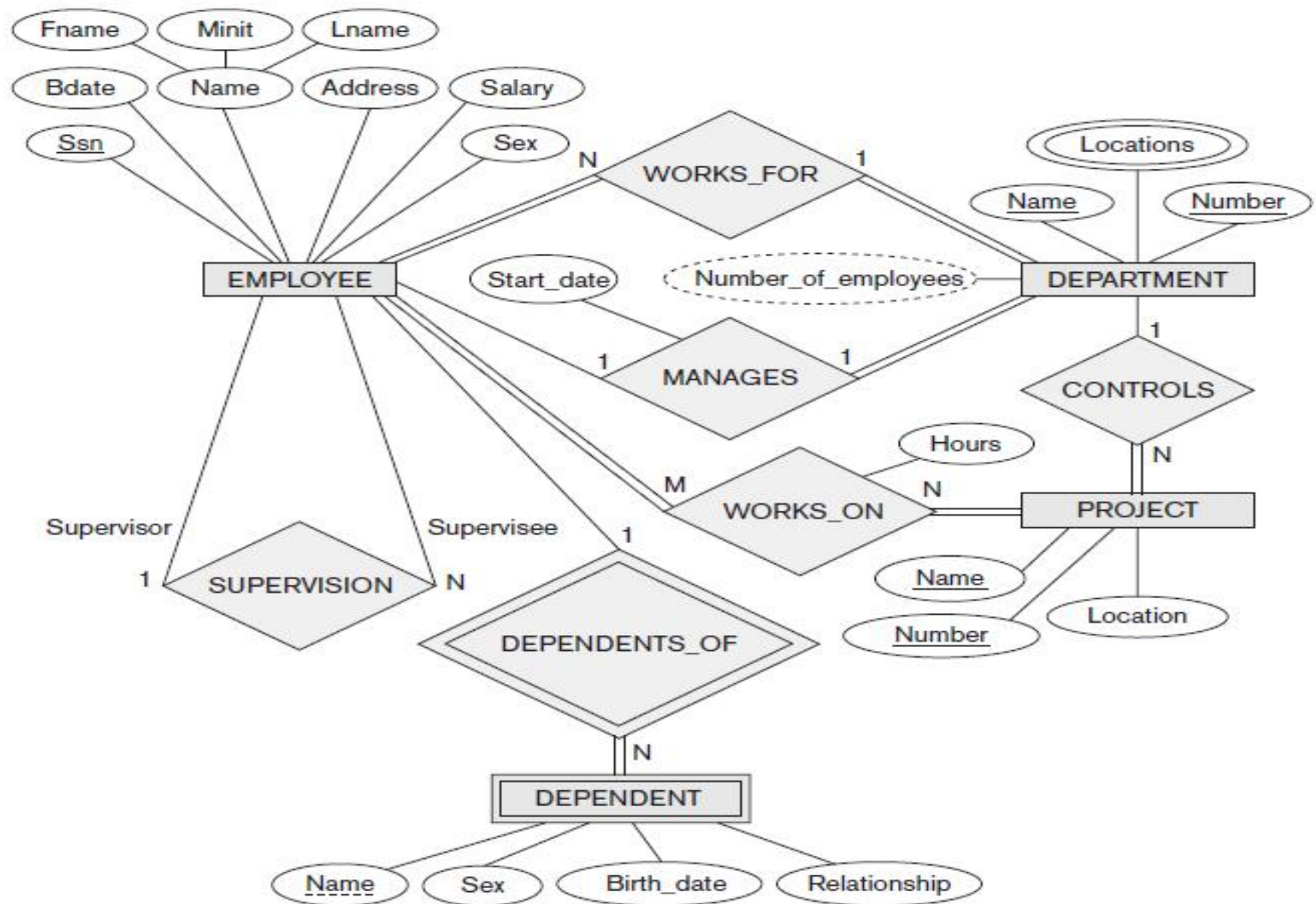
<u>License_number</u>	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 5.4

The CAR relation, with
two candidate keys:
License_number and
Engine_serial_number.

Relational Database Schema

- **Relational Database Schema:**
 - A set S of relation schemas that belong to the same database.
 - S is the name of the whole **database schema**
 - $S = \{R_1, R_2, \dots, R_n\}$ and a set IC of integrity constraints.
 - $R_1, R_2, \dots, R_n$ are the names of the individual **relation schemas** within the database S
- Following slide shows a COMPANY database schema with 6 relation schemas



COMPANY Database Schema

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Figure 5.5
Schema diagram for
the COMPANY
relational database
schema.

Relational Database State

- A **relational database state** DB of S is a set of relation states $DB = \{r_1, r_2, \dots, r_m\}$ such that each r_i is a state of R_i and such that the r_i relation states satisfy the integrity constraints specified in IC.
- A relational database *state* is sometimes called a relational database *snapshot* or *instance*.
- We will not use the term *instance* since it also applies to single tuples.
- A database state that does not meet the constraints is an invalid state

Populated database state

- Each *relation* will have many tuples in its current relation state
- The *relational database state* is a union of all the individual relation states
- Whenever the database is changed, a new state arises
- Basic operations for changing the database:
 - INSERT a new tuple in a relation
 - DELETE an existing tuple from a relation
 - MODIFY an attribute of an existing tuple
- Next slide (Fig. 5.6) shows an example state for the COMPANY database schema shown in Fig. 5.5.

Populated database state for COMPANY

Figure 5.6

One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Entity Integrity

- **Entity Integrity:**

- The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of $r(R)$.
 - This is because primary key values are used to *identify* the individual tuples.
 - $t[PK] \neq \text{null}$ for any tuple t in $r(R)$
 - If PK has several attributes, null is not allowed in any of these attributes
- Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity

- A constraint involving **two** relations
 - The previous constraints involve a single relation.
- Used to specify a **relationship** among tuples in two relations:
 - The **referencing relation** and the **referenced relation**.

Referential Integrity

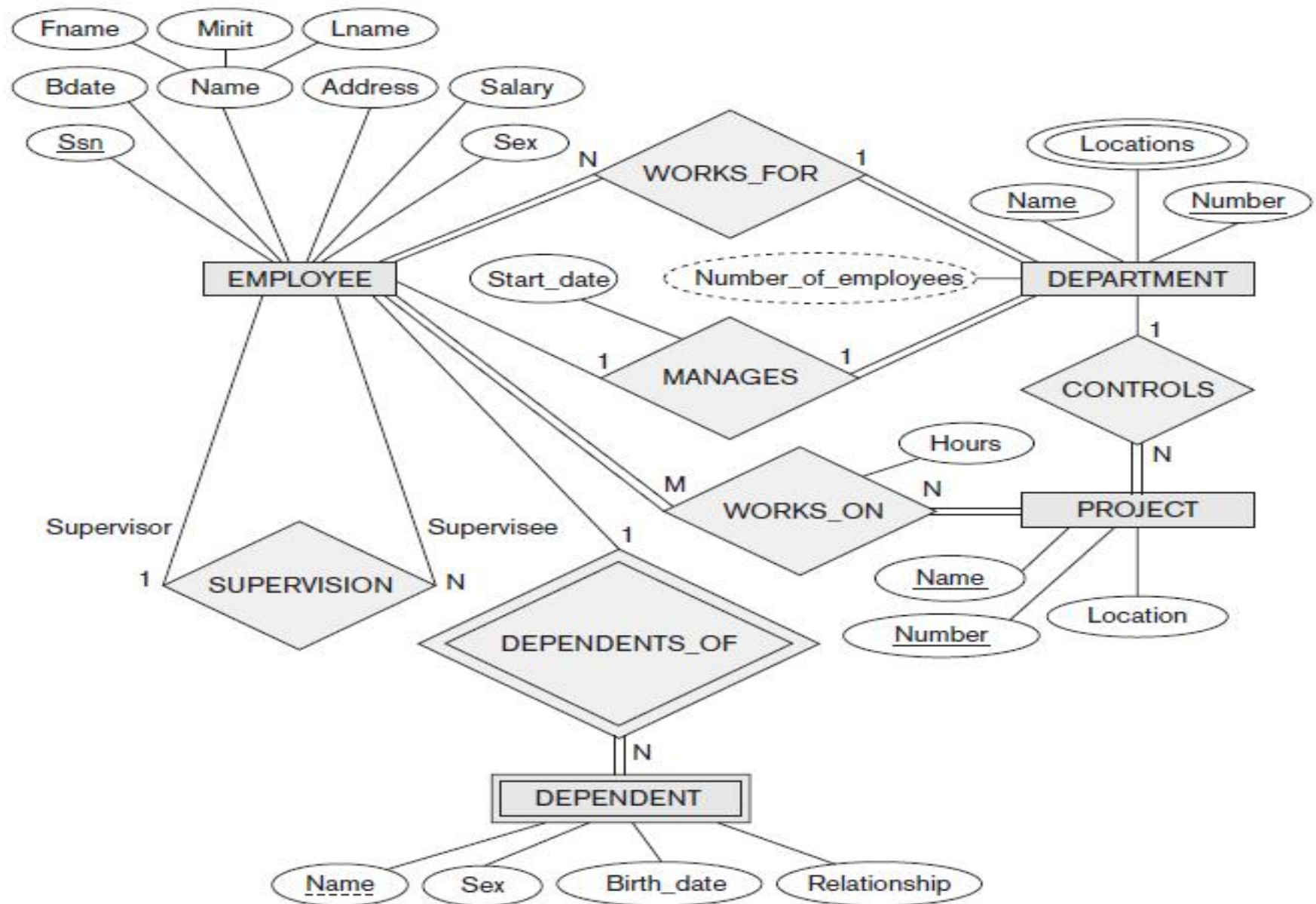
- Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R2.
 - A tuple t1 in R1 is said to **reference** a tuple t2 in R2 if $t1[FK] = t2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

Referential Integrity (or foreign key) Constraint

- Statement of the constraint
 - The value in the foreign key column (or columns) FK of the the **referencing relation** R1 can be **either**:
 - (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, or
 - (2) a **null**.
- In case (2), the FK in R1 should **not** be a part of its own primary key.

Displaying a relational database schema and its constraints

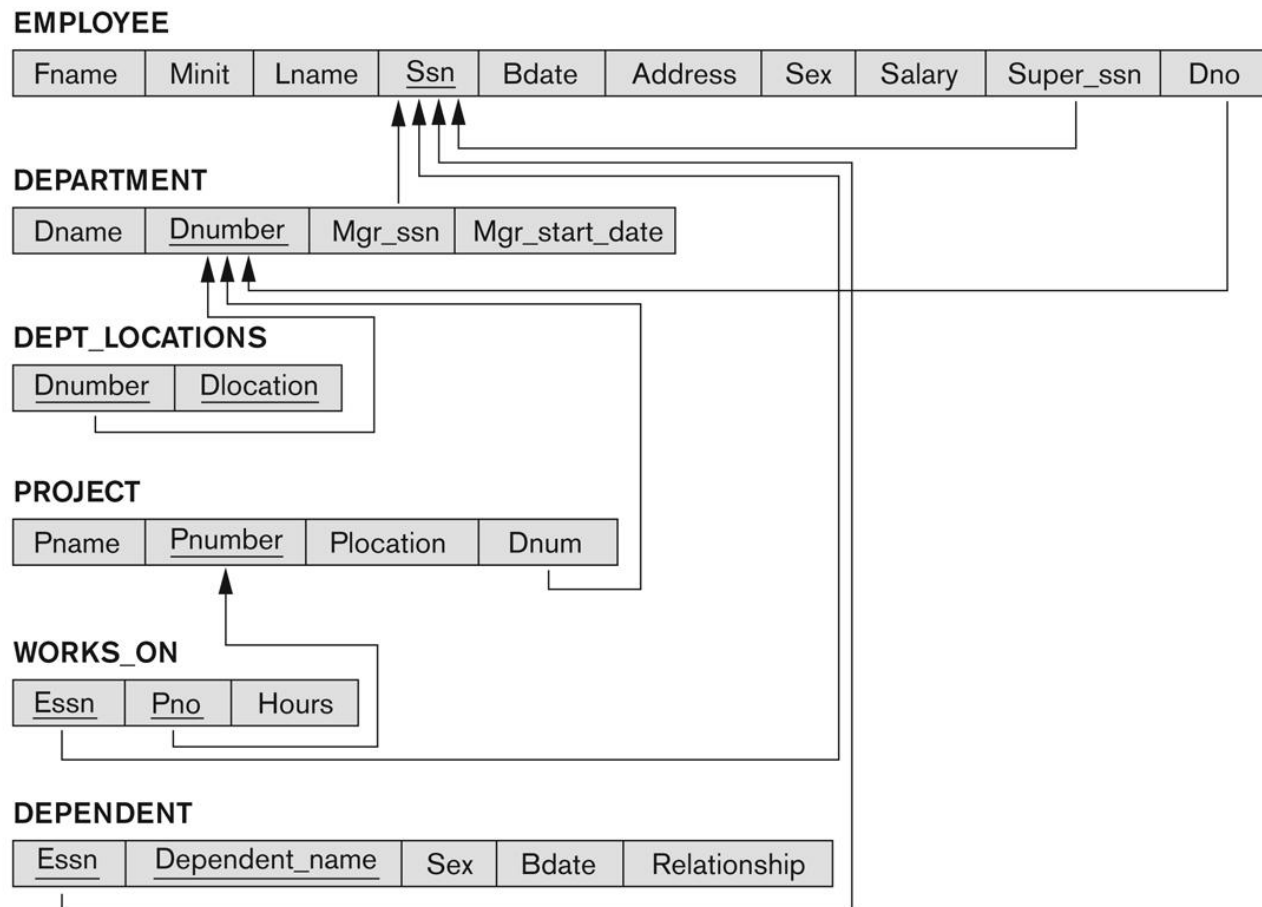
- Each relation schema can be displayed as a row of attribute names
- The name of the relation is written above the attribute names
- The primary key attribute (or attributes) will be underlined
- A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the referenced table
 - Can also point the the primary key of the referenced relation for clarity
- Next slide shows the COMPANY relational schema diagram with referential integrity constraints



Referential Integrity Constraints for COMPANY database

Figure 5.7

Referential integrity constraints displayed on the COMPANY relational database schema.



Other Types of Constraints

- Semantic Integrity Constraints:
 - based on application semantics and cannot be expressed by the model per se
 - Example: “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
- A **constraint specification** language may have to be used to express these
- SQL-99 allows **CREATE TRIGGER** and **CREATE ASSERTION** to express some of these semantic constraints
- Keys, Permissibility of Null values, Candidate Keys (Unique in SQL), Foreign Keys, Referential Integrity etc. are expressed by the **CREATE TABLE** statement in SQL.

Referential Integrity Constraints for COMPANY database

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

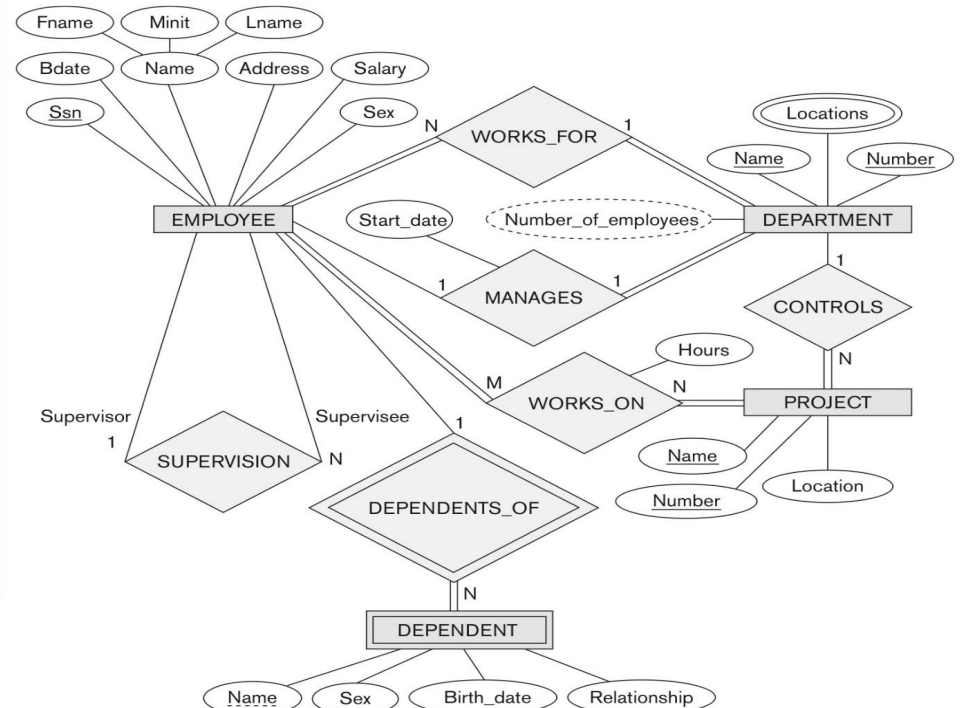
Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------



ER and EER -to-Relational Mapping

■ ER-to-Relational Mapping Algorithm

- Step 1: Mapping of Regular Entity Types
- Step 2: Mapping of Weak Entity Types
- Step 3: Mapping of Binary 1:1 Relation Types
- Step 4: Mapping of Binary 1:N Relationship Types.
- Step 5: Mapping of Binary M:N Relationship Types.
- Step 6: Mapping of Multivalued attributes.
- Step 7: Mapping of N-ary Relationship Types.

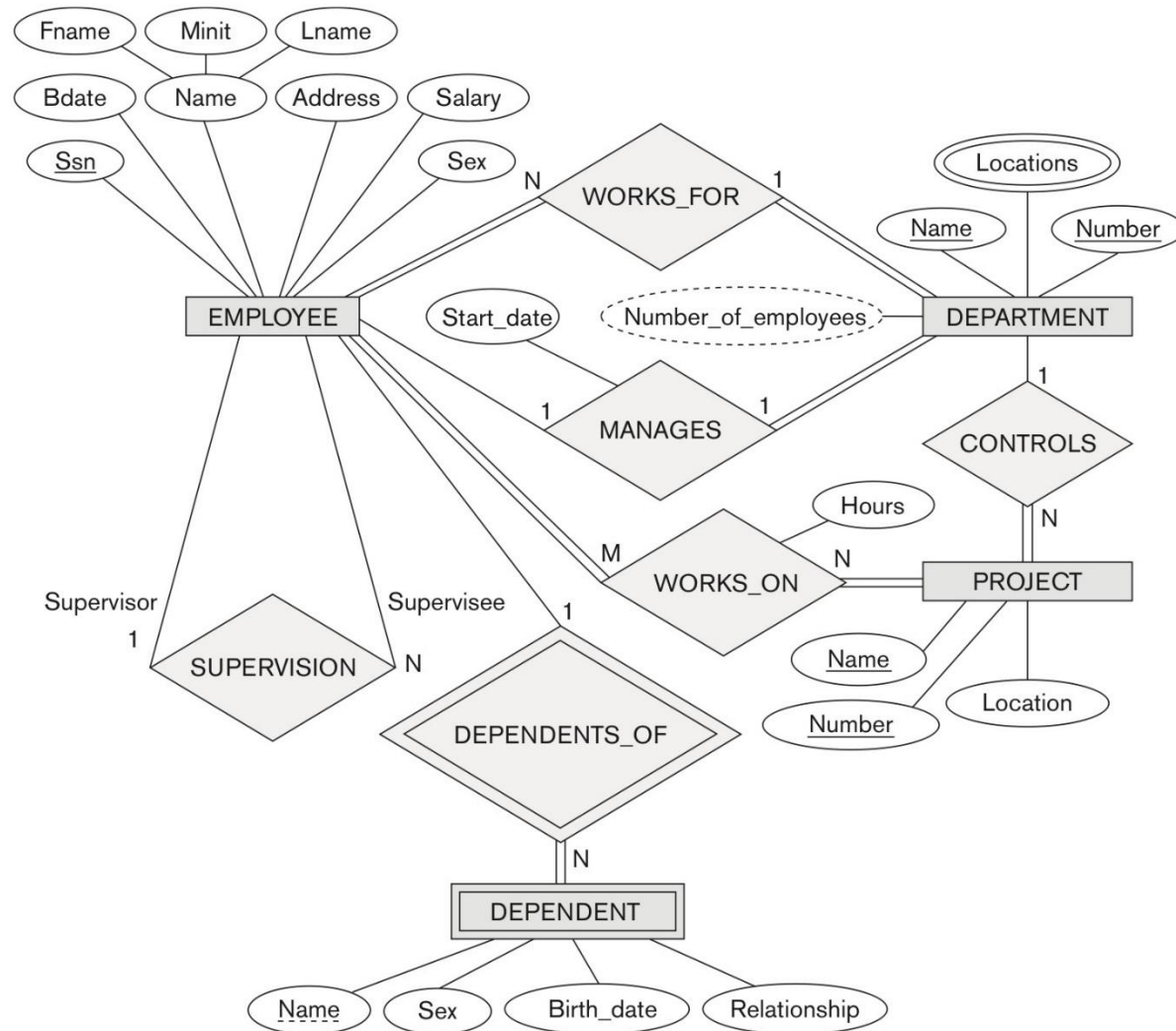
■ Mapping EER Model Constructs to Relations

- Step 8: Options for Mapping Specialization or Generalization.
- Step 9: Mapping of Union Types (Categories).

ER-to-Relational Mapping Algorithm

- Step 1: Mapping of Regular Entity Types.
 - For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
 - Choose one of the key attributes of E as the primary key for R.
 - If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.
- Example: We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram.
 - SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.

Figure 9.1 The ER conceptual schema diagram for the COMPANY database.



ER-to-Relational Mapping Algorithm (contd.)

■ Step 2: Mapping of Weak Entity Types

- For each weak entity type *W* in the ER schema with owner entity type *E*, create a relation *R* & include all simple attributes (or simple components of composite attributes) of *W* as attributes of *R*.
- Also, include as foreign key attributes of *R* the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- The primary key of *R* is the *combination of* the primary key(s) of the owner(s) and the partial key of the weak entity type *W*, if any.

■ **Example:** Create the relation **DEPENDENT** in this step to correspond to the weak entity type **DEPENDENT**.

- Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN).
- The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

ER-to-Relational Mapping Algorithm (contd.)

■ Step 3: Mapping of Binary 1:1 Relation Types

- For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
- There are three possible approaches:
 1. **Foreign Key (2 relations) approach:** Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.
 - Example: 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.
 2. **Merged relation (1 relation) option:** An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.
 3. **Cross-reference or relationship relation (3 relations) option:** The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

ER-to-Relational Mapping Algorithm (contd.)

- Step 4: Mapping of Binary 1:N Relationship Types.
 - For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
 - Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
 - Include any simple attributes of the 1:N relation type as attributes of S.
- Example: 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure.
 - For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.
- An alternative approach is to use a Relationship relation (cross referencing relation) – this is rarely done.

ER-to-Relational Mapping Algorithm (contd.)

- **Step 5: Mapping of Binary M:N Relationship Types.**
 - For each regular binary M:N relationship type R, *create a new relation S* to represent R. This is a *relationship relation*.
 - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; *their combination will form the primary key* of S.
 - Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.
- **Example: The M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema.**
 - The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed PNO and ESSN, respectively.
 - Attribute HOURS in WORKS_ON represents the HOURS attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

ER-to-Relational Mapping Algorithm (contd.)

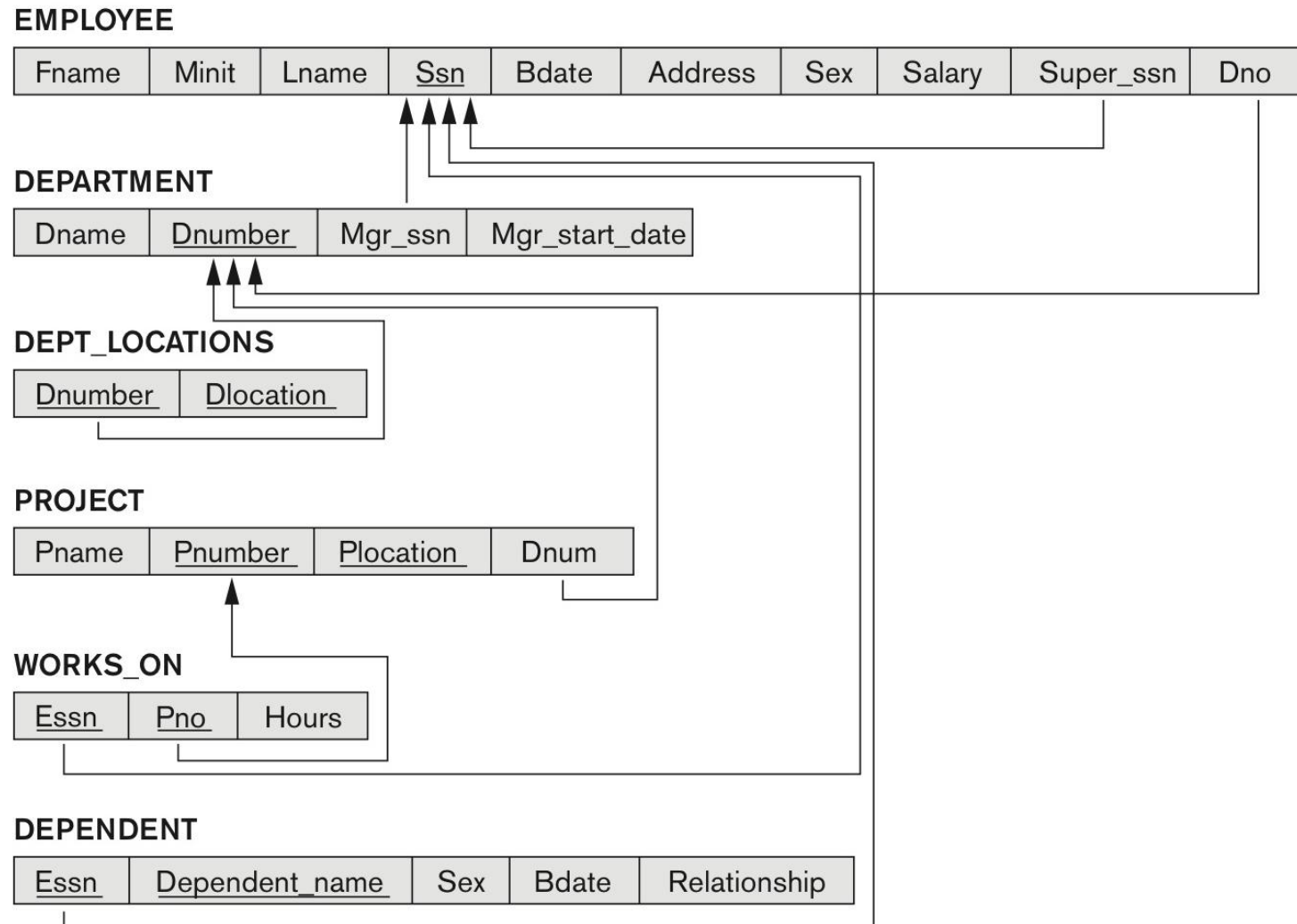
■ Step 6: Mapping of Multivalued attributes.

- For each multivalued attribute A, create a new relation R.
- This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
- The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

■ Example: The relation DEPT_LOCATIONS is created.

- The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation.
- The primary key of R is the combination of {DNUMBER, DLOCATION}.

Figure 9.2 Result of mapping the COMPANY ER schema into a relational database schema.

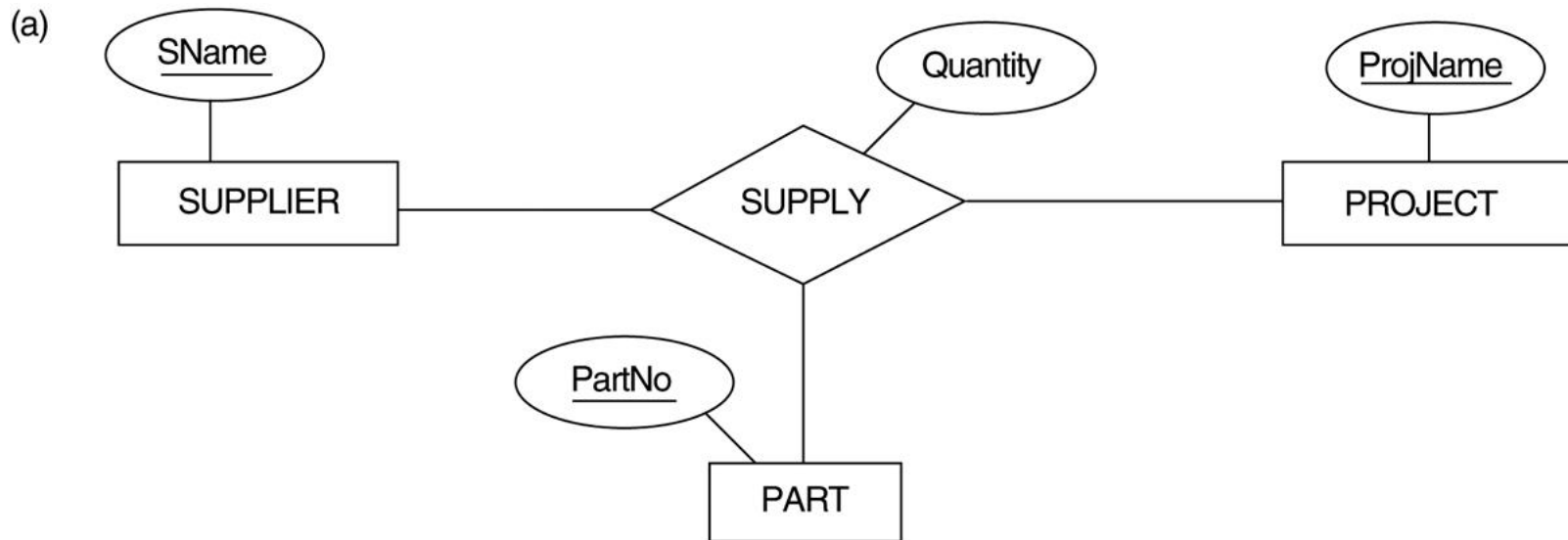


ER-to-Relational Mapping Algorithm (contd.)

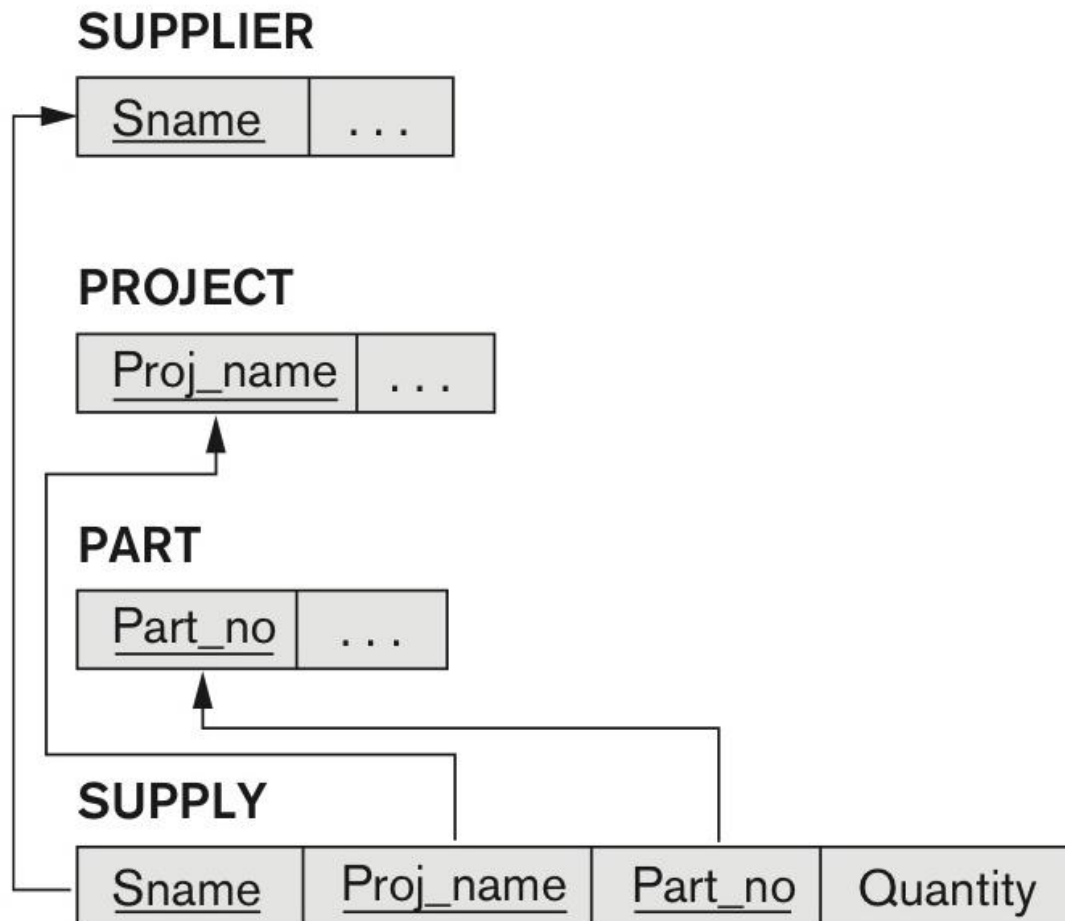
- **Step 7: Mapping of N-ary Relationship Types.**
 - For each n-ary relationship type R, where $n > 2$, create a new relationship S to represent R.
 - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
 - Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.
- **Example:** The relationship type SUPPY in the ER on the next slide.
 - This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME}

FIGURE 3.17

TERNARY RELATIONSHIP: SUPPLY



Mapping the *n*-ary relationship type SUPPLY



Summary of Mapping constructs and constraints

Table 9.1 Correspondence between ER and Relational Models

ER MODEL	RELATIONAL MODEL
Entity type	<i>Entity</i> relation
1:1 or 1:N relationship type	Foreign key (or <i>relationship</i> relation)
M:N relationship type	<i>Relationship</i> relation and <i>two</i> foreign keys
<i>n</i> -ary relationship type	<i>Relationship</i> relation and <i>n</i> foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key